(1)

Fault Type Enumeration and Classification

M. McElvany Hugue*

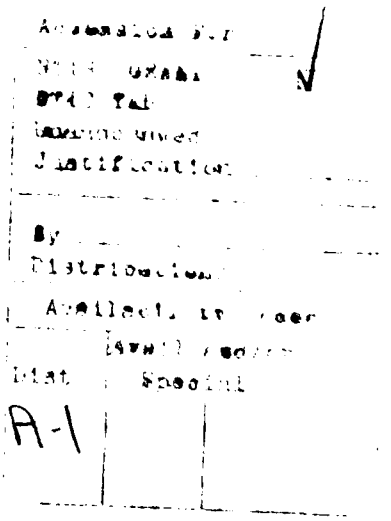
ONR-910915-MCM-TR9105

November 11, 1991

*Supported by ONR Contract # N00014-91-C-0014

EXECUTIVE SUMMARY

This report surveys of fault taxonomies used in classifying faults, and enumerates many commonly used fault types. Faults can be classified according to many attributes including cause, origin, persistence, duration, nature, extent, value, activity, symmetry and malice. After defining these and other fault attributes, we classify some fault types, and discuss the difficulties encountered in classifying many faults according to these taxonomies without sufficient knowledge of the characteristics of the environments, systems, and applications in which the faults arise. We enumerate and define faults in three classes: physical, environmental and operational faults. Finally, we present a total fault classification to be used within systems to define the classes of faults of interest and to specify the measures taken to tolerate those faults.



Contents

List of Figures

List of Tables

1 Introduction

In this report, we present several fault classification taxonomies and an enumeration of fault types.[1] The original goal of this document was to establish a comprehensive fault taxonomy, to enumerate all types of faults, and to classify all faults according to that taxonomy. However, after presenting classes based on the fault attributes and enumerating many fault types, we encountered difficulties in mapping specific faults to attributes. We could not classify all enumerated faults strictly according to the attributes, because we found that the assignment of a specific fault to a given class depends upon the characteristics of the system in which the fault occurs, as well as on the system environment and on the technology used to implement the system. Instead, we provide charts to be used in classifying faults in the context of the system, application, and environment of interest. A total fault classification approach is also provided, which indicates one way of specifying the techniques needed to handle a given fault and of demonstrating that those techniques are implemented.

We first discuss existing fault classification taxonomies, their associated fault attributes. We also include attributes based on fault effects. After we enumerate many commonly discussed faults, we attempt to map them to the fault attributes derived in the first section. We provide examples illustrating the difficulty of fault classification in the absence of any consideration of the fault environment, and explain the charts that are provided as a guide to system design. We also define the total fault classification to permit specification of the fault toleration methods that are implemented. After describing the impact of various combinations of attributes on reliability modeling, we discuss the limitations of this work, and indicate the direction to be taken to alleviate these limitations.

2 Classical Fault Taxonomy I

Many different fault classifications have been proposed. Laprie [1] defines the classes *accidental, intentional, physical, human-made, internal, external, design, operational, permanent and temporary* faults, described from the viewpoints of *nature, origin, and persistence*, as shown in Table 2, and defined in Table 1. He further presents likely combinations of these attributes, shown in Figure 1, with the symbol "$\star$" indicating the attributes being combined.[2] The common labeling of these combinations as faults is given at the right of the row. Table 3 enumerates the fault types *physical, transient, intermittent, design, interaction, malicious logic, and intrusion faults*, as defined by Laprie.

While these are classifications and attributes which are commonly used in the literature, they do not furnish all the information about faults that is required by the designers or assessors of a system. They provide little indication of the types of errors that are caused.

[1] A *fault* is the identified or hypothesized cause of an error. An *error* is the manifestation of a fault, an undesired state either at the boundary or at an internal point in the system. A *failure* is the inability of the system or component to provide the specified service caused by an error.

[2] $\star$ means that the combination can occur. Later, we use ∇ to indicate combinations which can't occur.

Nature
- Accidental Faults
- Intentional Faults

Origin
- Cause
  - Physical Faults
  - Human-made Faults
- System Bounds
  - Internal Faults
  - External Faults
- Phase of Creation
  - Design Faults
  - Operational Faults

Persistence
- Permanent
- Temporary

Table 1: Laprie Fault Hierarchy

Accidental faults	occur or created fortuitously
Intentional faults	occur or created deliberately
Physical faults	result of adverse physical phenomena
Human-made faults	caused by human imperfections
Internal faults	occur within a system
External faults	result of environmental interference
Design faults	human-made internal fault
Operational faults	occur during use of the system
Permanent faults	internal or external, unrelated to pointwise system conditions, of indefinite duration
Temporary faults	present in system for limited time
Transient faults	temporary external physical fault

Table 2: Laprie Fault Class Definitions

Physical faults	accidental, physical, internal, permanent and operational faults.
Intermittent faults	accidental, temporary, internal and either operational or design, physical or human made.
Design faults	accidental, human made, internal, permanent and design faults
Interaction faults	accidental, human-made,external, operational and temporary.
Malicious logic faults	intentional, human-made, internal, design and either permanent or temporary.
Intrusions	intentional, human-made, external, operational and either permanent or temporary

Table 3: Laprie Combination Fault Types

or of how to avoid, detect, or tolerate the faults. For example, the separation of human-made faults from design faults does not indicate where in the design process fault-avoidance techniques could be used to lessen such human mistakes.

3 Classical Fault Taxonomy II

The next fault taxonomy, adapted from [2, 3], describes faults based on the attributes of *cause, nature, duration, extent, value and activity*, described below and shown in Figure 2, with brief definitions in Table 4. The classes within each attribute are disjoint; so, for a given attribute, a specific fault *in a specific system* can belong to exactly one class. However, if the same type of fault occurs in two different systems, it is possible for the fault to be classified differently, with respect to that attribute, in each system. Such fault classification differences arise based on differences in the the systems in which it occurs. These differences can be due to where the fault occurs in the systems, how long the fault is active relative to the time scale (mission time) for each system, the technology used to construct each system, the assumed system environment, and other factors. Unlike the classification of Laprie, this taxonomy partitions human-made faults into sets identified by their causes, providing insight into where fault avoidance techniques can be used to prevent these mistakes.

3.1 Fault Cause

Fault causes include *specification mistakes, implementation mistakes, external disturbances* and *component defects.*

Specification mistakes cause the system or component described in the specification to differ from one which would provide the desired service correctly. They include typographical

Nature		Origin						Persistence		Type
		Cause		System Bounds		Phase of Creation				
Accidental Faults	Intentional Faults	Physical Faults	Human-made Faults	Internal Faults	External Faults	Des Faults	Oper Faults	Perm	Temp	
*		*		*			*	*		Physical
*		*			*		*		*	Transient
*		*		*			*		*	Intermittent (1)
*			*	*		*			*	Intermittent (2)
*			*	*		*		*		Design
*			*		*		*		*	Interaction
	*		*	*		*		*		Malicious Logic (1)
	*		*	*		*			*	Malicious Logic (2)
	*		*		*		*	*		Intrusions (1)
	*		*		*		*		*	Intrusions (2)

Figure 1: Laprie Fault Class Combinations

errors in a document, omission of important requirements, inclusion of conflicting requirements, incorrect requirements, and mistaken environmental assumptions.

Implementation mistakes include misinterpretation of a specification, design mistakes, manufacturing problems and integration problems. Hardware and software design mistakes may also be implementation mistakes.

External disturbances include environmental changes, EMI (electromagnetic interference), EME (high energy electromagnetic environments), temperature, vibrations, HERF (high energy radio frequency field), RF (low energy radio frequency field), radiation, EMP (electromagnetic pulse, nuclear or lightning direct strike), and SEU (single event upset due to high-energy particles such as α particles). Human mistakes such as mechanical construction problems, unexpected inputs, and other system misuse are also classed as external disturbances.

Component defects include physio-chemical disorders, threshold changes, manufacturing problems, short circuits, open circuits, bridging (input and feedback) faults, stuck-at faults, crosspoint or contact faults, and stuck open faults, as well as damage, fatigue or other deterioration. In the case of new technologies such as CMOS/SOS, gallium arsenide, room-temperature superconductors, and VHSIC/VLSIC, the possible component defects are not even fully documented.

3.2 Fault Nature

Fault nature discerns between *hardware* and *software* faults. While *hardware* faults can arise from any and all of the four causal factors described previously in §3.1, *software* faults are strictly from specification and implementation (programming) mistakes. Note that it may not always be easy to decouple hardware faults from software faults. For example, a software fault which causes a specific variable to (incorrectly) maintain a value of "1" is indistinguishable from a stuck-at-1 hardware fault at the address in which that variable is stored, unless an appropriate diagnostic routine exists and is successful in discerning that the hardware is not stuck at 1.

3.3 Fault Duration

Fault duration specifies how long the fault is active in the system. *Permanent* faults exist indefinitely. *Transient* faults can appear and disappear very quickly. *Intermittent* faults appear, disappear and may reappear repeatedly. This is the fault model most commonly used in reliability modeling, as it determines the success of various fault recovery strategies in restoring system services.

3.4 Fault Extent

The fault extent specifies whether the fault is *local*, affecting only a given hardware or software module, or, *global*, affecting multiple hardware or software modules, or both. Note that the fault extent may be limited using good design techniques to avoid and to contain faults. The term *fault scope* is often used to mean the extent of a fault.

3.5 Fault Value

The fault value can be *determinate*, where the fault status is unchanged over time unless externally acted upon, or *indeterminate*, where the fault status may change over time. These are also referred to as *stationary* and *non-stationary* faults, respectively.

3.6 Fault Activity

The fault activity specifies whether or not the fault is *active* or *dormant*. *Active* faults are present in the system or component, and are generating errors. *Dormant* faults are present, but are not generating errors; typically, they are in a component or a section of software not being exercised.

It should be noted that the term *latent fault* is often used to refer to both dormant faults and active faults whose errors escape detection. For this reason, we do not include latent faults as a separate class to maintain the mutual exclusion among classes within a given attribute.

Figure 3 contains a chart which indicates probable fault combinations, similar to that shown in Figure 1. The table can be read by row or by column, indicating the likelihood that a given fault is in the classes given by both the row and the column. Impossible or improbable combinations are indicated by "∇", probable combinations by "$\star$", and possible combinations are left blank.

Note: This chart differs from that in Figure 1, because in Figure 1, the row indicates a specific type of fault. As is clear from the presence of two different types of Intermittent, Malicious Logic, and Intrusion faults in Figure 1, a specific fault occurring in a specific system is not classified as belonging to two mutually exclusive fault classes simultaneously. In Figure 3, a different philosophy is used. All a "$\star$" in this table shows is that it is probable that a single instantiation of a fault can be placed in the classes given by the row and column simultaneously. While a blank entry indicates that it is possible for a single fault to be simultaneously in the classes given by the row and column, no attempt is made to enumerate all possible combinations of fault classes. Similarly, a "∇" means that it is highly unlikely that a specific fault could be placed in the classes given by the row and column. So, if the row and column entry represents two fault classes that are known to be mutually exclusive, such as hardware and software, the corresponding entry should be marked by a "∇". Also, a "$\star$" or a blank entry in this chart in Figure 3 *should not* be interpreted as indicating that a fault in the row class could cause a fault in the column class not to be

Cause
- Specification Mistakes
- Implementation Mistakes
- Specification Mistakes
- External Disturbances

Nature
- Hardware
- Software

Duration
- Permanent
- Intermittent
- Transient

Extent
- Local
- Global

Value
- Determinate
- Indeterminate

Activity
- Active Faults
- Dormant Faults

Figure 2: Fault Taxonomy

Symbol	Name	Definition
SM	Specification Mistake	design mistake, incorrect requirements, typographic errors
IM	Implementation Mistake	design mistake, manufacturing problem, damage, fatigue
ED	External Disturbance	harsh environments, EMI, radiation, unexpected inputs, system misuse
CD	Component Defect	physical imperfection or flaw in a component. Can be caused by any of design mistakes, manufacturing problems, damage, fatigue, deterioration.
HW	Hardware fault	arise from combination of SM, IM, ED, CD.
SW	Software fault	specification, programming implementation mistakes, or unexpected inputs (SM, IM, ED)
PT	Permanent fault	exists indefinitely
IT	Intermittent fault	appears, disappears and may reappear repeatedly
TT	Transient fault	appear and disappear very quickly
LO	Local fault	affects only a given hardware or software module (HW, SW)
GL	Global fault	affects multiple hardware, software modules
DT	Determinate fault	value unchanged over time
IN	Indeterminate fault	value changes over time
AV	Active fault	producing errors
DR	Dormant fault	present in system, but not producing errors

Table 4: Fault Attributes

tolerated, nor vice versa. The remainder of the charts in this paper should be interpreted similarly. We make this point clearer in the following example.

Example 1 As an example, consider the entries corresponding to implementation mistake (IM) faults. Due to the mutually exclusive partitions of the *cause* attribute, it is impossible for an IM fault to also be a specification mistake (SM), external disturbance (ED), or a component defect (CD). While an SM fault could lead to an IM fault, they must be treated as separate faults. So, the symbol "∇" indicates the impossibility of these fault combinations. Since IM faults could be either hardware (HW) or software (SW), these entries are left blank. Similarly, depending on the location of the IM fault, it could be permanent (PM), transient (TT), or intermittent (IT); local (LO) or global (GL); determinate (DT) or indeterminate (IN); and, active (AV) or dormant (DR). So, all possible IM faults could be classified by taking one entry from each sets of parentheses below:

$$\text{(IM) (SW, HW) (PT, IT, TT) (LO, GL) (DT, IN)(AV, DR)},$$

giving

$$1 \times 2 \times 3 \times 2 \times 2 \times 2 \ = 48.$$

So, there are 48 different possible IM fault variations using this taxonomy.□

As the previous example shows, fault classification based on solely on these attributes, while accurate and possible, is not practical because there is not enough specific information about the system to prune the fault set. Instead, this chart, and the others included in this document, should be used as guides to system design and analysis. Based on the system specifications, the target application, the technology used in constructing the components, the environment in which the system is to operate, and any other mitigating factors, certain fault combinations can immediately be ruled out. Similarly, certain design decisions, such as the use of fault containment regions (FCR's), which partition the system in such a way that any faults in separate FCR's can be treated as independent, can serve to limit the types of faults that can occur. Thus, for the remainder of this document, we will classify faults only to the extent described in the note above: "∇" for impossible or improbable classifications, "$\star$" for the most probable classification, and a blank entry to indicate a possible classification.

While the fault taxonomy presented in this section indicates where fault avoidance methods can be used, it is still incomplete because it only considers single faults, and does not address how the faults are manifested as errors. Thus, we introduce yet another set of attributes.

4 Additional Attributes

Several other attributes have been proposed to characterize behavior not directly addressed by the taxonomies previously described. These include *count*, *malice* and *symmetry*, described below. They are included in the chart given in Figure 6 for operational faults.

Attribute	Cause				Nature		Duration			Extent		Value		Activity	
Fault	Spec Mis (SM)	Impl Mis (IM)	Ext Dist (ED)	Comp Dfct (CD)	Hard ware (HW)	Soft ware (SW)	Perm nant (PT)	Intr mtnt (IT)	Tran sient (TT)	Loc al (LO)	Gbl al (GL)	Det nate (DT)	Indet nate (IN)	Act ive (AV)	Dorm ant (DR)
SM		∇	∇	∇											
IM	∇		∇	∇											
ED	∇	∇		∇											
CD	∇	∇	∇												
HW						∇									
SW					∇										
PT								∇	∇						
IT							∇		∇						
TT							∇	∇							
LO											∇				
GL										∇					
DT													∇		
IN												∇			
AV															∇
DR														∇	

Figure 3: Fault Attribute Combinations
(Key: ⋆—probable, ∇—not probable)

4.1 Fault Count—Multiple or Single Faults

The fault *count* indicates whether single or multiple faults occur, an issue not addressed directly by the previous taxonomies, but of significant interest in reliability modeling. Fault *time* is usually used in the context of multiple faults, which can be coincident[3] or separated in time. Fault *source*, also in the context of multiple faults, discerns between *independent* and *correlated* faults.

4.2 Fault Symmetry and Malice

The attributes of symmetry and malice describe how the fault is manifested as errors within a system or component. These attributes have been combined to produce reliability estimates in [4] in the context of interactive consistency algorithms. Before defining these attributes, we introduce terminology describing how faults appear within a system or component and how faults are tolerated.

Active or dynamic redundancy attempts to achieve fault-tolerance by fault-detection alone, or in conjunction with location and recovery.[2] Active redundancy techniques include duplication with comparison; standby sparing, either hot or cold; pair and a spare, as well as information redundancy using data encoding or range and sanity checks.

Passive redundancy uses voting mechanisms to mask fault occurrences. [2] Fault masking is used in passive redundancy techniques to hide the occurrence of faults and to eliminate the effects of the faults, i.e., to avoid *errors*. Passive approaches are usually transparent to the user or operator, and, in their simplest form, make no attempt to detect the fault, much less its source. Passive techniques include use of triple-modular redundancy (TMR), where the hardware is tripled; and N-modular redundancy, where N hardware modules are used. For further details, see [2].

The *scope* of a fault refers to the area of the system or component where the fault causes errors, or to the extent of a fault. For example, the scope of a global fault (GL) could be the entire system, or could be all hardware modules of a given type; the scope of a local fault (LO) could be a single hardware module, a single component, or a single processor. Fault scope can be limited using good design techniques.

Using these definitions, we define the fault attributes of *malice* and *symmetry* based on attributes of the errors generated by a fault.

4.2.1 Fault Malice

A fault can be either *malicious* or *non-malicious*. A *non-malicious fault* produces errors that can be detected using an active redundancy technique. A *malicious fault* produces errors that cannot be detected using active redundancy techniques, but require masking using a passive redundancy technique.

[3]Coincident faults may be "nearly coincident", with a second fault occurring before recovery from the first has completed

4.2.2 Fault Symmetry

Faults can also be either *symmetric* or *asymmetric*. A *symmetric fault* generates errors that are manifested identically throughout the scope of the fault, *i.e.*, the portion of the system or component affected by the fault. An *asymmetric fault* generates errors that are manifested differently throughout the scope of the fault.

5 Enumeration of Faults

Having described these fault classes, we next enumerate faults to be classified. Many types of faults have been recognized, with a representative subset listed below, organized into sets based on their origins or effects, as reflected in the names used for the sets, *physical, environmental,* and *operational.* [4]

5.1 Physical Faults

Faults in this set arise from either component defects, or external disturbances under the *cause* attribute. A chart to be used to classify these faults for specific applications is supplied in Figure 4, with the abbreviations used in the chart given in boldface in Table 5. Fault definitions are taken from [5] and [6].

5.2 Environmental Faults

Environmental faults are caused by conditions external to the system or component. This set includes EMI (electromagnetic interference), EME (high energy electromagnetic environments), temperature, vibrations, HERF (high energy radio frequency) fields, RF (low energy radio frequency) fields, radiation, EMP (electromagnetic pulse, nuclear or lightning direct strike), and SEU (single event upset due to high-energy particles such as α particles). Human mistakes such as mechanical construction problems, unexpected inputs, and other system misuse are also environmental faults. Some environmental faults are enumerated in Table 6.

5.3 Operational Faults

The next set of faults is categorized by their effects on the service delivered or on system operation. A list is given in boldface in Tables 7 and 8. The fault classes defined by the attributes of count, duration, malice, and symmetry in §4 are included in this set due to their effects on system operations.

[4] The set names were chosen to indicate either the fault origins or effects, and should not be confused with any of the fault attributes previously described.

Symbol	Name	Definition
SA0	stuck at 0	bit stuck at logical 0, in register, memory location, etc.
SA1	stuck at 1	bit stuck at logical 1
SAX	stuck at X	X is indeterminate value or voltage, may be interpreted differently by different receivers
SIG	signal line open	can look like stuck at fault in TTL logic, but not necessarily in MOS
GDO	ground open	can look like stuck at fault in TTL logic, but not necessarily in MOS
SVS	signal and V_{cc} shorted	can look like stuck at fault in TTL logic, but not necessarily in MOS
SSL	single stuck at line	an interconnection line stuck at 0, 1, or an indeterminate value
MSL	multiple stuck at line	several lines stuck at 0, 1, or an indeterminate value simultaneously
BR	bridging fault	permanent fault, in which two leads in a logic network are connect, called "wired logic" at the connection.
SS	Short	looks like stuck at in TTL, not in MOS technology
GF	gate fault	any fault involving a logic gate
SO	stuck open	physical fault causing the output of a gate to depend on present and previous inputs. In CMOS, not equivalent to stuck at fault.
MCH	mechanical	breakage, severing of communication lines, physical damage
CUP	coupling fault	fault occurring mainly in RAMS, where state transitions in one memory cell affect other cells

Table 5: Physical Faults

Attribute	Cause				Nature		Duration			Extent		Value		Activity	
Fault	Spec Mis (SM)	Impl Mis (IM)	Ext Dist (ED)	Comp Dfct (CD)	Hard ware (HW)	Soft ware (SW)	Perm nant (PT)	Intr mtnt (IT)	Tran sient (TT)	Loc al (LO)	Gbl al (GL)	Det nate (DT)	Indet nate (IN)	Act ive (AV)	Dorm ant (DR)
SA0		⋆		⋆									∇		
SA1		⋆		⋆									∇		
SAX		⋆		⋆									∇		
SIG		⋆		⋆		∇						∇			
GDO		⋆		⋆		∇									
SVS						∇									
SSL		⋆		⋆		∇									
MSL				⋆		∇									
BR				⋆		∇									
SS				⋆		∇									
GF				⋆		∇									
SO				⋆		∇									
CUP					⋆	∇									
MCH			⋆			∇									

Figure 4: Physical Faults
(Key: ⋆—probable, ∇—*not probable*)

Symbol	Name
EME	High energy electromagnetic environments
EMI	Electromagnetic interference
EMP	Electromagnetic pulse
HERF	High energy radio frequency field
RF	Low energy radio frequency field
LG	Lightning
RAD	Radiation
ALF	Alpha particles
PS	Power spike
SEU	Single event upset
PSF	Power supply fluctuation

Table 6: Environmental Faults

Attribute	Cause				Nature		Duration			Extent		Value		Activity	
Fault	Spec Mis (SM)	Impl Mis (IM)	Ext Dist (ED)	Comp Dfct (CD)	Hard ware (HW)	Soft ware (SW)	Perm nant (PT)	Intr mtnt (IT)	Tran sient (TT)	Loc al (LO)	Gbl al (GL)	Det nate (DT)	Indet nate (IN)	Act ive (AV)	Dorm ant (DR)
EME			⋆			∇									
EMI			⋆			∇									
EMP			⋆			∇									
HERF			⋆			∇									
RF			⋆			∇									
LG			⋆			∇									
RAD			⋆			∇									
ALF			⋆			∇									
TEMP			⋆												
PS			⋆	⋆			∇								
PSF			⋆	⋆			∇								
SEU															∇

Figure 5: Environmental Faults
(Key: ⋆—probable, ∇—not probable)

Symbol	Name	Definition
OM	omission	fault in which an expected service is not delivered
CAT	catastrophic	fault whose consequences greatly exceed the benefits of proper system service
SPF	single point of failure	a component, module or function whose loss causes the system to fail
BEN	benign	faults consisting of omission of messages, or delays in sending or relaying messages [7]
ARB	arbitrary	any type of fault, from benign through unrestricted
BYZ	Byzantine	ARB, MAL fault, intentionally attempting to defy detection and interfering with any attempt at tolerance [8]
MAL	malicious	fault which escapes detection by active redundancy, also ARB and BYZ
AUTH	authenticated	restricted fault, with initial behavior ARB or MAL, but communication requires unforgeable messages, which limits further arbitrary behavior.[8, 9]
NMAL	non-malicious	can be detected using active redundancy [4]
SYM	symmetric	generates identical errors to all components, subsystems, etc. within the scope of the fault [4]
ASYM	asymmetric	generates different values, error and correct possible, to all components, etc. within the scope of the fault
HARD	hard	fault requiring actions to be taken to prevent activation; also, solid fault.
SOFT	soft	fault for which no measures are taken to prevent activation

Table 7: Operational Faults I

Attribute	Cause				Nature		Duration			Extent		Value		Activity	
Fault	Spec Mis (SM)	Impl Mis (IM)	Ext Dist (ED)	Comp Dfct (CD)	Hard ware (HW)	Soft ware (SW)	Perm nant (PT)	Intr rntnt (IT)	Tran sient (TT)	Loc al (LO)	Gbl al (GL)	Det nate (DT)	Indet nate (IN)	Act ive (AV)	Dorm ant (DR)
OM															
CAT			⋆				⋆								
SPF															
BEN															
ARB															
BYZ															
MAL															
AUTH															
NMAL															
SYM													∇		
ASYM															
SOFT															
HARD															

Figure 6: Operational Fault Enumeration I
(Key: ⋆—probable, ∇—not probable)

Symbol	Name	Definition
GEN	generic	design fault, occurring in common modules, can be MCOI, CM, and, arguably, SPF
CM	common mode	affects a set of components or modules due to a faulty shared resource, such as a common power supply, or to a fault in a hardware or software module used by a set of components
STA	stationary	fault whose value is consistent over time; also called determinate fault
NSTA	non-stationary	fault whose value may vary over time; also called indeterminate fault
MCOI	multiple, coincident	two or more faults occurring at the same time, or, in a system with recovery, a second fault occurs before the first can be handled properly, called near-coincident faults.
MSEP	multiple, separated in time	two or more faults occur, but the problems of MCOI and near-coincident faults are avoided. If recovery is used in the system, the first fault is correctly handled before the second occurs.
MIND	multiple, independent	2 or more coincident or near-coincident faults, but different modules, etc.
MCOR	multiple, correlated	MCOI related to each other, either in same physical location, same component type, same circuits. Could be CM, or due to damage.
ACT	active	fault producing errors
LAT	latent	ACT, but not recognized as such, or DOR
DOR	dormant	fault not activated by computation process, can be either software, in code not exercised, or in hardware, in perhaps a cold spare
PER	permanent	fault existing indefinitely
TRN	transient	fault appearing and disappearing quickly
INT	intermittent	fault appears, disappears, and may reappears, but effect may not be repeatable easily
TIM	timing	fault causing timing of service delivery to vary from the specification

Table 8: Operational Faults II

Attribute	Cause				Nature		Duration			Extent		Value		Activity	
Fault	Spec Mis (SM)	Impl Mis (IM)	Ext Dist (ED)	Comp Dfct (CD)	Hard ware (HW)	Soft ware (SW)	Perm nant (PT)	Intr mtnt (IT)	Tran sient (TT)	Loc al (LO)	Gbl al (GL)	Det nate (DT)	Indet nate (IN)	Act ive (AV)	Dorm ant (DR)
GEN															
CM															
STA													∇		
NSTA												∇			
MCOI															
MSEP															
MIND															
MCOR															
ACT															∇
LAT															
DOR														∇	
PER									∇	∇					
INT							∇		∇						
TRN							∇	∇							
TIM															

Figure 7: Operational Fault Enumeration II
(Key: ⋆—probable, ∇—not probable)

We next use the fault classes defined in §3 to organize the faults we have enumerated in this section.

6 Mapping Specific Faults to Attribute Fault Classes

In general, specific fault-to-class assignments depend upon the system, the technology used in implementing the system, the application environment and requirements, and other factors, as demonstrated for fault class combinations in Example 1. For example, suppose we consider a general fault f. Depending upon the location and extent of the fault, either global or local, the effects could be either permanent, transient or intermittent. If a part of the hardware not currently being used is damaged by the fault, the fault is dormant; otherwise, it's active. Similarly, the fault can be determinate or indeterminate. We cannot choose a random enumerated fault type and claim that it is always in a given class in all systems in which it occurs. Thus, the charts shown in Figures 4-7 should be used as guidelines in the design or evaluation of a given system. For each of the fault sets enumerated in § 5, we choose a specific fault and explain the considerations to be used in classifying faults within this framework.

6.1 Physical Faults

Most physical faults are the result of either component defects (CD) or external disturbances (ED), as is evident from the prevalence of *stars* ($\star$'s) in the columns corresponding to CD and ED in Figure 4. Another factor making classification difficult is the the potential inability to distinguish between two types of faults having the same effect, as demonstrated in the following example.

Example 2 Suppose we consider the classification of a stuck-at-1 (SA1) fault. A software (SW) fault which causes a specific variable to (incorrectly) maintain a value of "1" is indistinguishable from a stuck-at-1 (SA1) hardware (HW) fault in the location in which that variable is stored, unless an appropriate diagnostic routine exists and is successful in discerning that the hardware is not stuck at 1. If the fault really is in the software, the duration of the "1" in that variable location might be transient (TT), if the section of software is only executed for a portion of the mission, but could also judged as permanent (PT) if that portion of the mission is long enough. If the software is swapped in and out, the fault might appear as an intermittent (IT) SA1, which sounds unlikely, but is obviously possible.□

Thus, the technology used to implement the system, the application, the design methodology used to contain faults, and other factors must be examined to classify potential faults according to their attributes.

6.2 Environmental Faults

Most environmental faults are caused by external disturbances (ED), as demonstrated by the $\star$'s in the column corresponding to ED in Figure 5. However, the reason that the system fails to tolerate them may also be a fault, making an exact classification or fault count difficult, as shown in the following example.

Example 3 Consider a radiation (RAD) or an alpha-particle (ALF) fault. Clearly, such a fault is caused by the environment (ED), but the inability of a system to tolerate such a fault could be due to a mistaken assumption in the specification (SM) that such faults can't occur, or in estimating the probability of such particles in the anticipated environment (SM). Or, the specification could have correctly anticipated such faults, but the implementation was incorrect (IM), or the components were not designed to be rad-hard, or they were design to be rad-hard, but a manufacturing difficulty arose (CD). The only thing we can say is that it is not possible for a RAD or ALF fault to affect software, except indirectly by altering the hardware in which the software resides. For that reason, the symbol "∇" appears in the boxes corresponding to (RAD, SW), and (ALF, SW) in Figure 4 to indicate that a software failure is independent of this type of environmental fault, and the boxes corresponding to (RAD, ED), and (ALF, ED) each contain the symbol "$\star$".$\square$

The fault classification in the previous example relies on the definitions of ALF, HW, and SW faults to conclude that an ALF fault cannot cause a SW fault. However, in practice, it may be difficult to discern a combined ALF and HW fault from a SW fault. Due to the complexity of classifying primary fault effects demonstrated in Example 1, we do not address secondary fault effects in these charts. However, for diagnosis, an analysis of combined fault effects may become necessary. We address this difficulty in §7.

6.3 Operational Faults

The classification guidelines for faults altering the system operations or service is shown in Figure 6 and Figure 7. As before, the ∇ indicates impossible or improbable classifications, the $\star$ indicates probable classification, and an empty box indicates that it is possible to classify the specific fault on the row into the class indicated by the column.

By definition, an arbitrary (ARB) fault can be classified into any combination of fault classes which is not self-contradictory, so all its entries are empty. A catastrophic (CAT) fault is most likely due to an external disturbance (ED); so, a $\star$ appears in the (CAT, ED) entry of the table.

Again, we see that the specification of the system, the technology used to implement the system, the application, the design methodology used to contain faults, the environment in which the system operates, and other factors must be examined to classify potential faults according to their attributes. If such additional information were available, the charts in Figure 6 and Figure 7 could then be filled out, providing guidelines in evaluating the fault coverage and protection of the specific system.

7 Total Fault Classification

The preliminary classification by attribute of the faults listed above was not entirely successful, due to the dependence of fault effects upon the architecture, implementation, and application in which the faults occurred. Another way to characterize the behavior of faults and the system response to them, while still controlling the "attribute-explosion," is to use a *total fault classification*. The total fault classifications enumerates the types of faults anticipated or recognized by the system and includes the attributes most relevant to the design, to the technology used in implementing various components, to the dependability requirements, and to the fault tolerance, detection, isolation, and recovery techniques.

The total fault classification is similar to the fault dictionaries used in fault detection experiments [2], but at a higher level of abstraction. Faults are divided into classes based on how they are manifested in the system. If two different faults cause identical effects, then the faults are indistinguishable, and will fall into the same fault class. However, an intermittent fault may fall into two different classes because the effects of the fault on the system might also be a function of the system state; so, the temporal separation of two occurrences of the fault might be counted as two different faults. The secondary fault effects mentioned in Example 3 must be addressed to derive the total fault classification.

For each fault class judged to be relevant to the system, application and implementation, the total fault classification should consist of the following information, to the extent that it is known or can be determined. An inability to specify how faults in a certain class are handled may mean that a system needs modifications to tolerate those faults.

1. **Description:** how to discern faults in this class, in terms of the fault, error, or failure in the system.

2. **Cause:** what precipitates faults in this class, actual or hypothesized (if known), such as design mistakes, damage, external disturbances, short circuit.

3. **Duration:** temporal effects, either permanent, transient or intermittent, where the scale used to discern between permanent or transient faults is quantified.[5]

4. **Malice:** how the fault is manifested throughout its scope. Non-malicious faults can be detected using active redundancy techniques, such as range checks, error-correcting codes, and comparison. Malicious faults defy detection by such techniques; toleration requires passive redundancy techniques.

5. **Symmetry:** symmetric or asymmetric. Symmetric faults are seen identically by all observers of the fault throughout the fault extent. Asymmetric faults may be mani-

[5] A "transient" stuck-at fault that lasts for 20 minutes and then subsides is permanent for a 15 minute mission with 100 millisecond control loops, while the same fault is transient for a system with mission times on the order of hours.

fested differently to different observers, with at least one pair of observers within the fault scope receiving differing results.

6. **Extent:** scope of the fault. Spatial effects, either local, global, or in between. Area of system or components effected by the fault.

7. **Criticality:** relative importance of this fault class to survivability, availability or any appropriate metric. A numerical classification may be used, or relative terms such as "essential," "flight critical," or "non-critical." A good operating definition of the terms in this classification must be included to ensure success of the classification.

8. **Containment strategy:** how the effects of faults are confined, if at all.

9. **Latency of detection:** maximum time between the *occurrence* of the fault and its detection. This may not be a useful bound, as the fault is usually detected when it causes errors. The error latency, the time between occurrence of an error and the detection of that error may be a better definition.

10. **Level of detection:** useful only if the system can be hierarchically partitioned with some faults handled at the level in the system at which they occur, and others handled at a higher or lower level. This includes global agreement algorithms used to mask or detect errors at a local level. Another level assignment could be signal, gate, chip, component, subsystem, system, etc. Or, hardware, operating system, network and communication levels could also be used.

11. **Detection strategy:** how fault is detected, e.g., using parity check, deviance checking, etc.

12. **Masking strategy:** how fault effects are masked, e.g., majority vote.

13. **Recovery latency:** maximum time between occurrence of a fault and recovery from the fault. Note that this includes detection latency.

14. **Recovery strategy:** how system state is restored or reset following the fault, e.g., retry, error correcting codes, forward and backward error recovery.

8 Conclusion—Impact on Reliability Modeling

While all the taxonomies and attributes described herein are valid, several are extremely useful in reliability modeling and in design and analysis. In modeling fault behavior and in classifying faults, the most important attributes are *malice*, *symmetry*, *duration*, and *count*.

Malice indicates the difficulty in tolerating the fault, and determines whether to use a passive or active redundancy technique to tolerate it. *Symmetry* indicates how the fault is manifested throughout the system, determining the type of passive redundancy technique

needed to tolerate the fault. So, if a fault is classified as malicious based on the types of errors it causes, but only active redundancy techniques are implemented in the system, then the system cannot be guaranteed to cover that fault should it be malicious. If the fault affects any critical components, the probability of that fault's malice, relative to the system reliability requirement, should be extremely small, or zero. Otherwise, the failure to use a passive redundancy technique to avoid a potential single point of failure will need to be justified.

Duration indicates how long the fault is active in the system. If the duration of a fault can somehow be detected, even if only through implication using the penalization with thresholds used in MAFT (the Multicomputer Architecture for Fault Tolerance) [10, 11], then the system may be able to handle transients and intermittents without failing otherwise healthy components.

Count is necessary because the calculation of reliability depends on the number of faults that a system or component can tolerate, as well as on the coincidence and independence of multiple faults.

While the total fault classification presented in the previous section may seem verbose, all the issues addressed in the classification must be considered in analyzing the behavior of faults in a system, if only to ensure that a specific issue has not been overlooked.

In future work, we will present models used to represent various fault types enumerated here. We will limit fault classification to the use of these four attributes, as the number of possible fault types then becomes manageable. We will refer to the other fault attributes as needed to distinguish among individual fault classes.

References

[1] J.-C. Laprie, "Dependability: Basic concepts and associated terminology." February 1990.

[2] B. W. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems.* Addison-Wesley Publishing Company, 1989.

[3] A. Avizienis and J.-C. Laprie, "Dependable computing: From concepts to design diversity," *Proceedings of the IEEE*, vol. 74, pp. 629–638, May 1986.

[4] P. Thambidurai and Y.-K. Park, "Interactive consistency with multiple failure modes," in *Proceedings, Seventh Symposium on Reliable Distributed Systems*, pp. 93–100, IEEE, October 1988.

[5] J. A. Abraham and W. K. Fuchs, "Fault and error models for VLSI," *Proceedings of the IEEE*, vol. 74, pp. 639–654, May 1986.

[6] J. P. Hayes, "Fault modeling," in *Fault-Tolerant Computing Tutorial* (V. P. Nelson and B. D. Carroll, eds.), pp. 37–44, 1730 Massachusetts Avenue, NW., Washington, D.C. 20036-1903: IEEE Computer Society Press, 1987.

[7] F. Meyer and D. Pradhan, "Consensus with dual failure modes," in *Proceedings, Seventeenth International Symposium on Fault Tolerant Computing*, pp. 48–54, IEEE, July 1987.

[8] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, pp. 382–401, July 1982.

[9] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *JACM*, vol. 27, pp. 228–234, April 1980.

[10] C. Walter, R. Kieckhafer, and A. Finn, "MAFT: A multicomputer architecture for fault-tolerance in real-time control systems," in *Proceedings, IEEE Real-Time Systems Symposium*, pp. 133–140, IEEE, December 1985.

[11] R. Kieckhafer, C. Walter, A. Finn, and P. Thambidurai, "The MAFT architecture for distributed fault tolerance," *IEEE Transactions on Computers*, vol. C-37, pp. 398–405, April 1988.